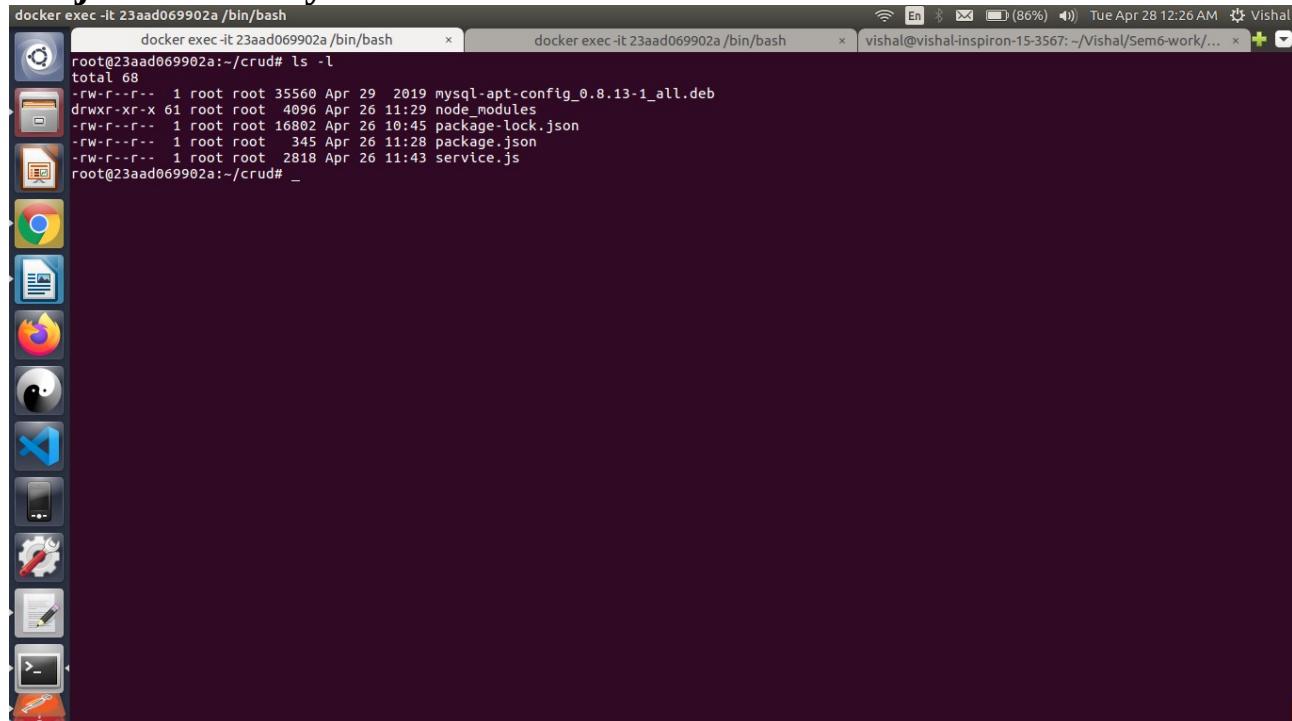


Roll Number:17134
Vishal Kokane
Assignment 002

This assignment is of building REST api using CRUD operations.
I've used nodejs +mysql in base docker image of ubuntu.

Git repository url:<https://github.com/vishalkokane264/pucsd2020-devops>

Project Directory structure:

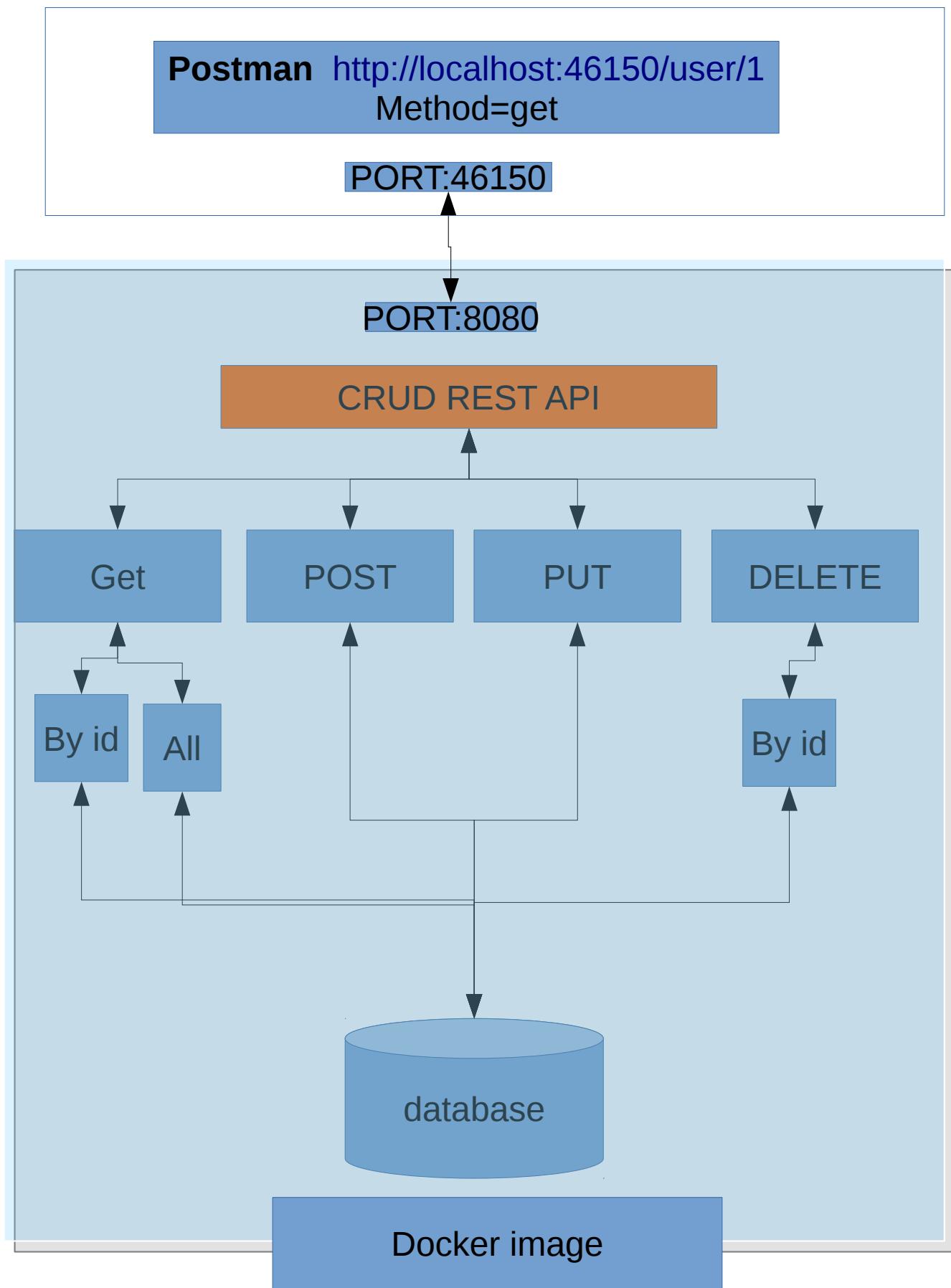


A screenshot of a Linux desktop environment. On the left, there's a dock with icons for various applications like a terminal, file manager, browser, and code editor. The main area shows two terminal windows. The first window is titled "docker exec -it 23aad069902a /bin/bash" and shows the command "ls -l" being run, displaying a directory listing with files like mysql-apt-config_0.8.13-1_all.deb, node_modules, package-lock.json, package.json, and service.js. The second window is titled "vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/..." and shows the command "ls" being run, displaying a list of files including .gitignore, Dockerfile, README.md, and requirements.txt.

Api Documentation:

Method	URL	Description
GET by id	http://localhost:46150/user/:id	Get user by user id
Get (all)	http://localhost:46150/user	Get all users
POST	http://localhost:46150/user	Add user in database
PUT	http://localhost:46150/user	Update username using userid
DELETE(id)	http://localhost:46150/user/id	Delete user using userid

Flow diagram



dockerfile:

```
FROM crud2:latest

RUN cd
RUN cd /root/crud
RUN /etc/init.d/mysql stop

RUN mysqld_safe --skip-grant-tables &

EXPOSE 8080

CMD [ "node", "index.js" ]
```

service.js:

```
// Including the file and initialize
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const mysql = require('mysql');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
    extended : true
}));

const mysql_conn = mysql.createConnection({
    host: 'localhost',
    user: 'vishal',
    database: 'userdb'
});
mysql_conn.connect();

app.get('/',function (req,res){
```

```
        return res.send({error:true, message:
"Rest service using CRUD ops."} );
} );

app.get('/user', function (req, res) {
    mysql_conn.query('SELECT * FROM user',
function (error, results, fields) {
        if (error) throw error;
        res.end(JSON.stringify(results));
    });
} );

// Retrieve user with id
app.get('/user/:id', function (req, res) {

    let user_id = req.params.id;

    if (!user_id) {
        return res.status(400).send({ error: true,
message: 'Enter userID' });
    }

    mysql_conn.query('SELECT * FROM tasks where
id=?', user_id, function (error, results, fields)
{
        if (error) throw error;
        return res.end(JSON.stringify(results));
    });
} );

// Add a new user
app.post('/user', function (req, res) {
    var userdata=req.body;
//    console.log(userdata);
    if (!req.body.id) {
```

```
        return res.status(400).send({ error:true,
message: 'Please enter userid' });
    }

    mysql_conn.query('INSERT INTO user
SET ?',userdata, function (error, results, fields)
{
    if (error) throw error;
    else{
        console.log("User added
successfully");
        res.end(JSON.stringify(results));
    }
});
});

// Update username with id
app.put('/user', function (req, res) {

    let user_id = req.body.id;
    let name = req.body.name;
//    console.log('my user id '+user_id+name);

    if (!user_id || !name) {
        return res.status(400).send({ error: task,
message: 'Enter user id and name' });
    }

    mysql_conn.query("UPDATE user SET name = ?
WHERE id = ?", [name, user_id], function (error,
results, fields) {
        if (error) throw error;
        return res.send({ error: false, data:
results, message: 'User updated successfully.' });
    });
});
```

```
// Delete user
app.delete('/user/:id', function (req, res) {

    let user_id = req.params.id;

    mysql_conn.query('DELETE FROM user WHERE id
= ?', [user_id], function (error, results, fields)
{
    if (error) throw error;
    return res.send({ error: false, data:
results, message: 'User deleted successfully.' });
});
});

// All other requests redirect to 404
app.all('*', function (req, res, next) {
    return res.send('Page not found');
next();
});

//Port must be set to 8080 because incoming http
request are routed from 80 to 8080

app.listen(8080, function(){
    console.log('App is running on port
8080');
});
```

package.json

```
{  
  "name": "express-node-rest-project",  
  "version": "1.0.0",  
  "description": "",  
  "main": "service.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test  
specified\\\" && exit 1"  
  },  
  "keywords": [ ],  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "body-parser": "^1.19.0",  
    "express": "^4.17.1",  
    "mysql": "^2.16.0"  
  }  
}
```

Snapshot:

Running Docker file

```

docker exec -it 23aad069902a /bin/bash
  docker exec -it 23aad069902a /bin/bash
  docker exec -it 23aad069902a /bin/bash
  vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/... + Visha

1588 docker build -t "mysql_ubuntu:dockerfile27" .
1589 docker build -t "mysql_ubuntu:dockerfile" .
1603 docker build -t "dockerbuild:dockerfile" .
1607 docker build -t "dockerbuild:dockerfile" .
1614 docker run -d -p 8070:8070 --name dockerbuild:dockerfile -e MYSQL_ROOT_PASSWORD=' dockerbuild:dockerfile
1623 docker build -t "dockerbuild:dockerfile" .
1669 docker build -t "dockerbuild:dockerfile" .
+ project2 git:(master) X docker build -t "crudrest:dockerfile" .
  Sending build context to Docker daemon 2.048kB
Step 1/7 : FROM crud2:latest
--> b4928ea4ac53
Step 2/7 : RUN cd
--> Running in 03655e63c121
  Removing intermediate container 03655e63c121
--> d63fdb8288
Step 3/7 : RUN cd /root/crud2
--> Running in 73eb6783ca28
  Removing intermediate container 73eb6783ca28
--> 807f5dca274c
Step 4/7 : RUN /etc/init.d/mysql stop
--> Running in e2d35ff52fd6
[Info] MySQL Community Server 5.7.29 is already stopped.
  Removing intermediate container e2d35ff52fd6
--> f14103dd9d64
Step 5/7 : RUN mysqld_safe --skip-grant-tables &
--> Running in 5996fbe7ecc3
  Removing intermediate container 5996fbe7ecc3
--> 962be8cb20a8
Step 6/7 : EXPOSE 8080
--> Running in bddef1db282
  Removing intermediate container bddef1db282
--> 1865fb7322b
Step 7/7 : CMD ["node","index.js"]
--> Running in d3c3983e4cf6
  Removing intermediate container d3c3983e4cf6
--> b58dbc12f077
Successfully built b58dbc12f077
Successfully tagged crudrest:dockerfile
+ project2 git:(master) X docker images
REPOSITORY          TAG        IMAGE ID      CREATED       SIZE
crudrest            dockerfile  b58dbc12f077  14 seconds ago  1.14GB
crud2               latest     b4928ea4ac53   4 minutes ago  1.14GB

```

1) Docker images && container running

```

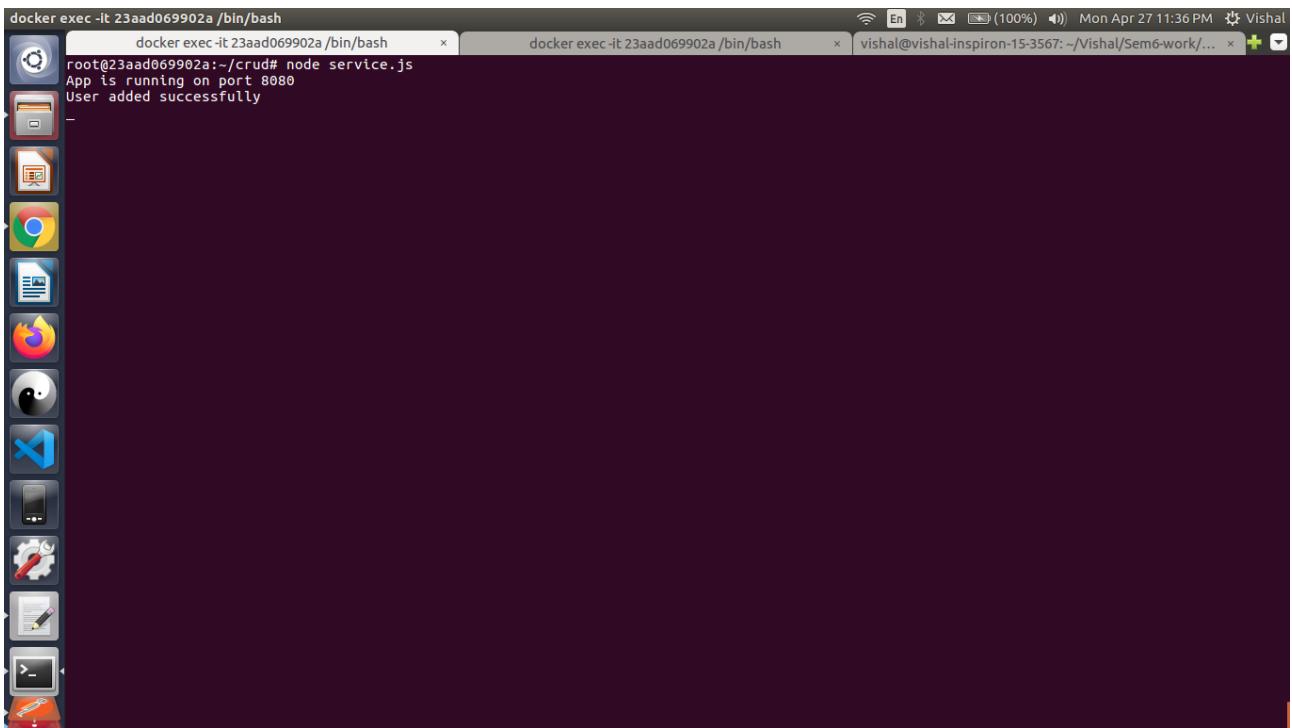
vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/pucsd2020-Tasks/Training/project2
  docker exec -it 23aad069902a /bin/bash
  docker exec -it 23aad069902a /bin/bash
  vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/... + Visha

+ project2 git:(master) X docker images
REPOSITORY          TAG        IMAGE ID      CREATED       SIZE
crudrest            dockerfile  b58dbc12f077  14 minutes ago  1.14GB
crud2               latest     b4928ea4ac53   18 minutes ago  1.14GB
vishal              latest     33ceed915f0c   2 hours ago   1.1GB
helloworld          latest     9dd3633380cc  2 hours ago   1.1GB
dockerbuild         dockerfile  53caa99b262  10 hours ago   1.1GB
new2                latest     a515a462ddd   10 hours ago   1.1GB
mysql-node-ubuntu   latest     a21d9eb750d8  11 hours ago   1.1GB
mysql_ubuntu         dockerfile  f9b78f8c5341  12 hours ago  644MB
crud-service         latest     e9a0cfb89b3d  30 hours ago  1GB
ubuntu              latest     1d622ef86b13  3 days ago    73.9MB
debian              latest     971452c94376  2 months ago   114MB
herreralutis/mysql-ubuntu latest   293d44326d32  3 years ago   345MB
+ project2 git:(master) X docker ps
CONTAINER ID        IMAGE           COMMAND       CREATED      STATUS      PORTS          NAMES
23aad069902a        b58dbc12f077   "/bin/bash"   13 minutes ago Up 13 minutes  0.0.0.0:46150->8080/tcp   crazy_goldwasser
+ project2 git:(master) X -

```

2) Nodejs service running on port 8080

```
docker exec -it 23aad069902a /bin/bash
root@23aad069902a:~/crud# node service.js
App is running on port 8080
User added successfully
-
```

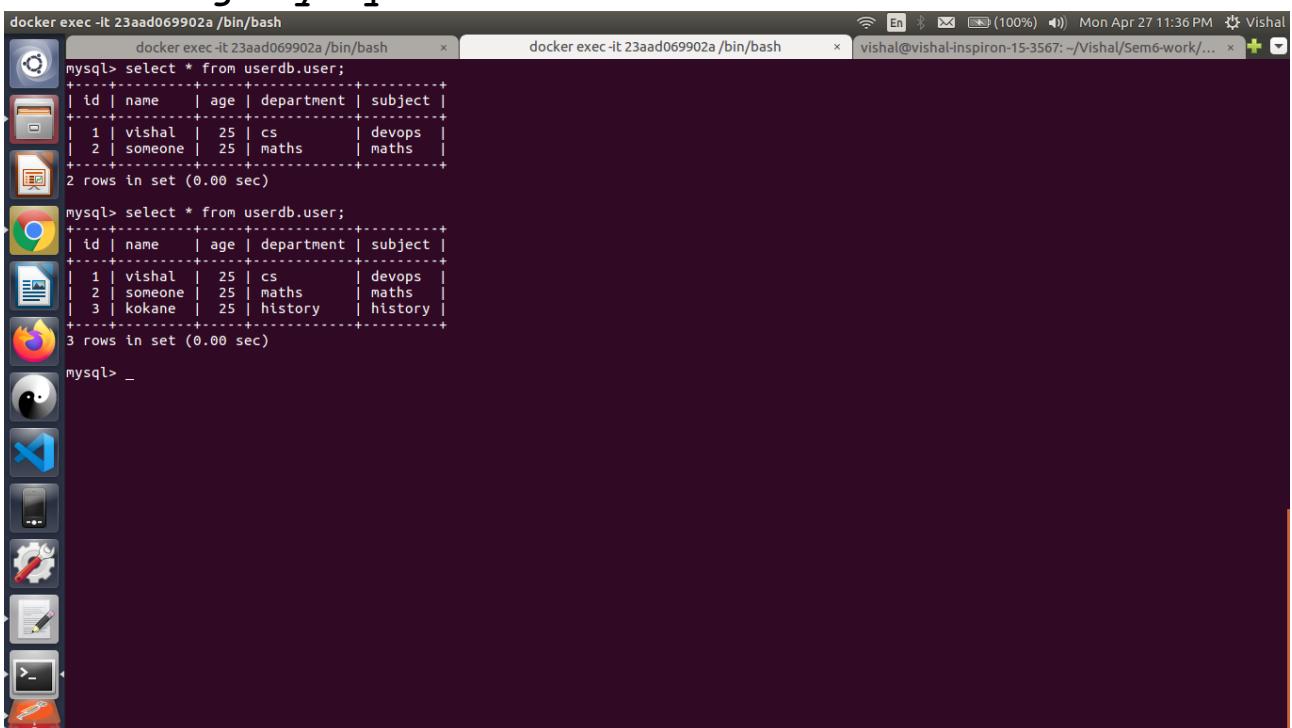


3) Another same container running for checking mysql data

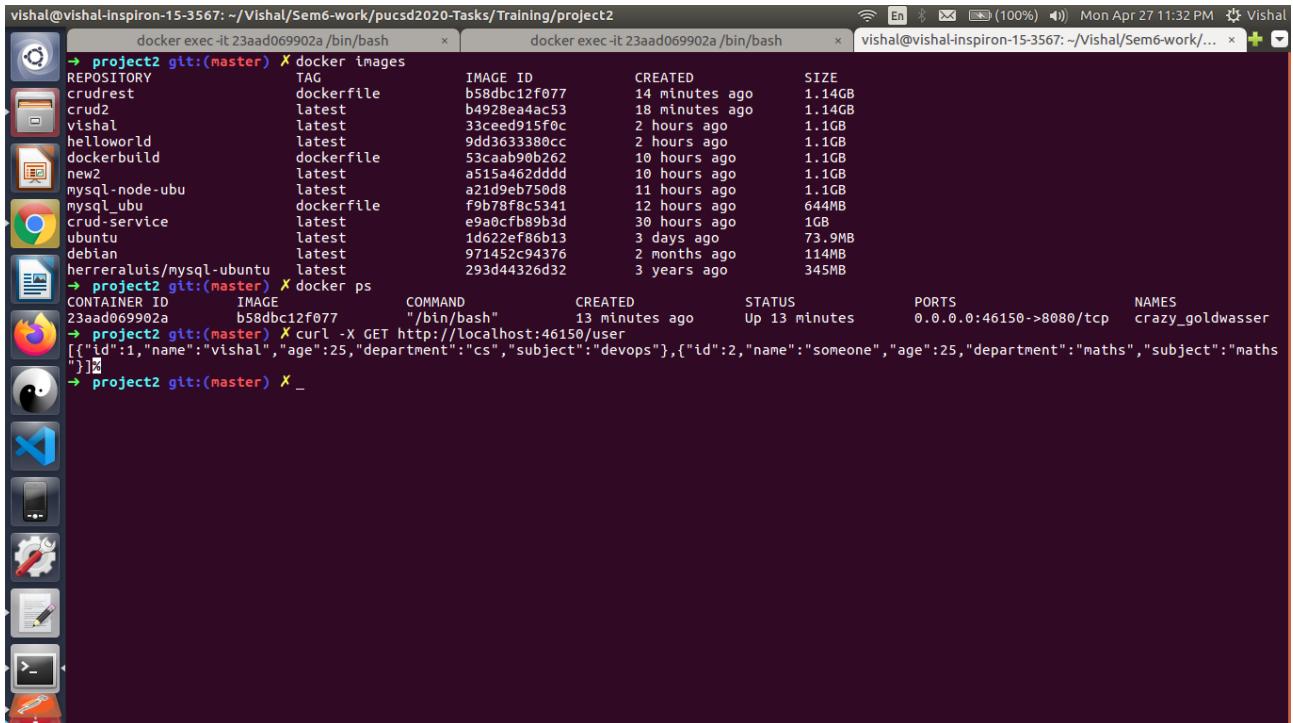
```
docker exec -it 23aad069902a /bin/bash
root@23aad069902a:~/crud# mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```



4)Validating service using CURL get request



```
vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/pucsd2020-Tasks/Training/project2
  docker exec -it 23aad069902a /bin/bash  docker exec -it 23aad069902a /bin/bash  vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/...
+ project2 git:(master) X docker images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
crudrest            dockerfile  b58dbc12f077  14 minutes ago  1.14GB
crud2               latest    b4928ea4ac53  18 minutes ago  1.14GB
vishal              latest    33ceed915f0c  2 hours ago   1.1GB
helloworld          latest    9dd3633380cc  2 hours ago   1.1GB
dockerbuild         dockerfile  53caab99b262  10 hours ago  1.1GB
new2                latest    a515a462ddd  10 hours ago  1.1GB
mysql-node-ubuntu   latest    f9b78f8c5341  12 hours ago  644MB
crud-service        latest    e9a0cfb89b3d  30 hours ago  1GB
ubuntu              latest    1d622ef86b13  3 days ago   73.9MB
debian              latest    971452c94376  2 months ago  114MB
herreraluis/mysql-ubuntu latest  293d44326d32  3 years ago   345MB
+ project2 git:(master) X docker ps
CONTAINER ID        IMAGE           COMMAND       CREATED      STATUS      PORTS          NAMES
23aad069902a        b58dbc12f077  "/bin/bash"   13 minutes ago  Up 13 minutes  0.0.0.0:46150->8080/tcp  crazy_goldwasser
+ project2 git:(master) X curl -X GET http://localhost:46150/user
[{"id":1,"name":"vishal","age":25,"department":"cs","subject":"devops"}, {"id":2,"name":"someone","age":25,"department":"maths","subject":"maths"}]
+ project2 git:(master) X _
```

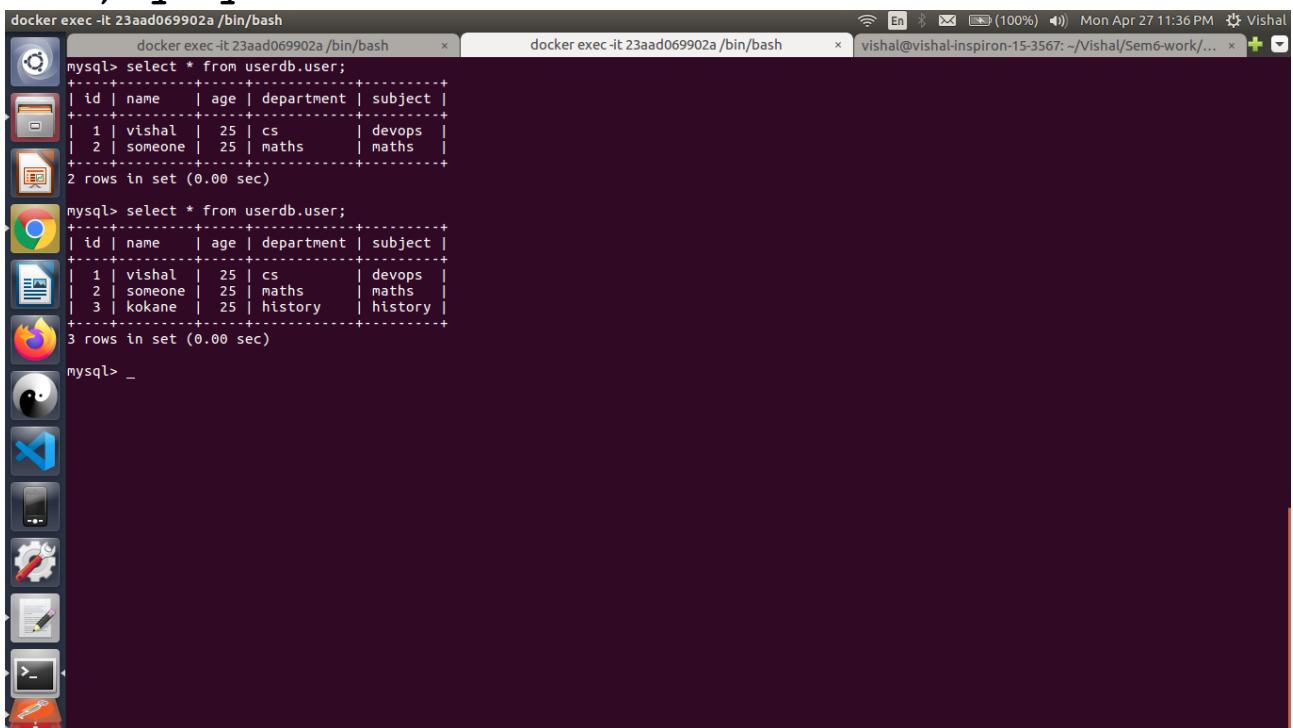
5) Postman POST method:

The screenshot shows the Postman application interface. In the top navigation bar, there are buttons for NEW, Runner, Import, and a search bar. To the right, it displays the date and time (Mon Apr 27 11:35 PM), the user's name (Vishal), and various status icons. The main workspace has a left sidebar titled 'History' showing a list of recent API calls, including POST requests to localhost:46150/user. The main panel shows a POST request to 'localhost:46150/user'. The 'Body' tab is selected, showing a JSON payload: {"id":3,"name":"kokane","age":25,"department":"history","subject":"history"}. Below the body, the response status is 200 OK with a time of 94 ms. The response body is also shown as JSON: {"fieldCount":0,"affectedRows":1,"insertId":0,"serverStatus":2,"warningCount":0,"message":"","protocol41":true,"changedRows":0}.

5.1) User added Successfully

The screenshot shows a terminal window with three tabs. The first tab shows the command 'docker exec -it 23aad069902a /bin/bash'. The second tab shows the command 'docker exec -it 23aad069902a /bin/bash' and the output 'root@23aad069902a:~/crud# node service.js'. The third tab shows the command 'vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work...' and the output 'User added successfully'. The terminal has a dark theme with a light-colored font. The left sidebar of the terminal window includes icons for file operations, browser, code editor, and terminal.

5.2) Mysql result of POST

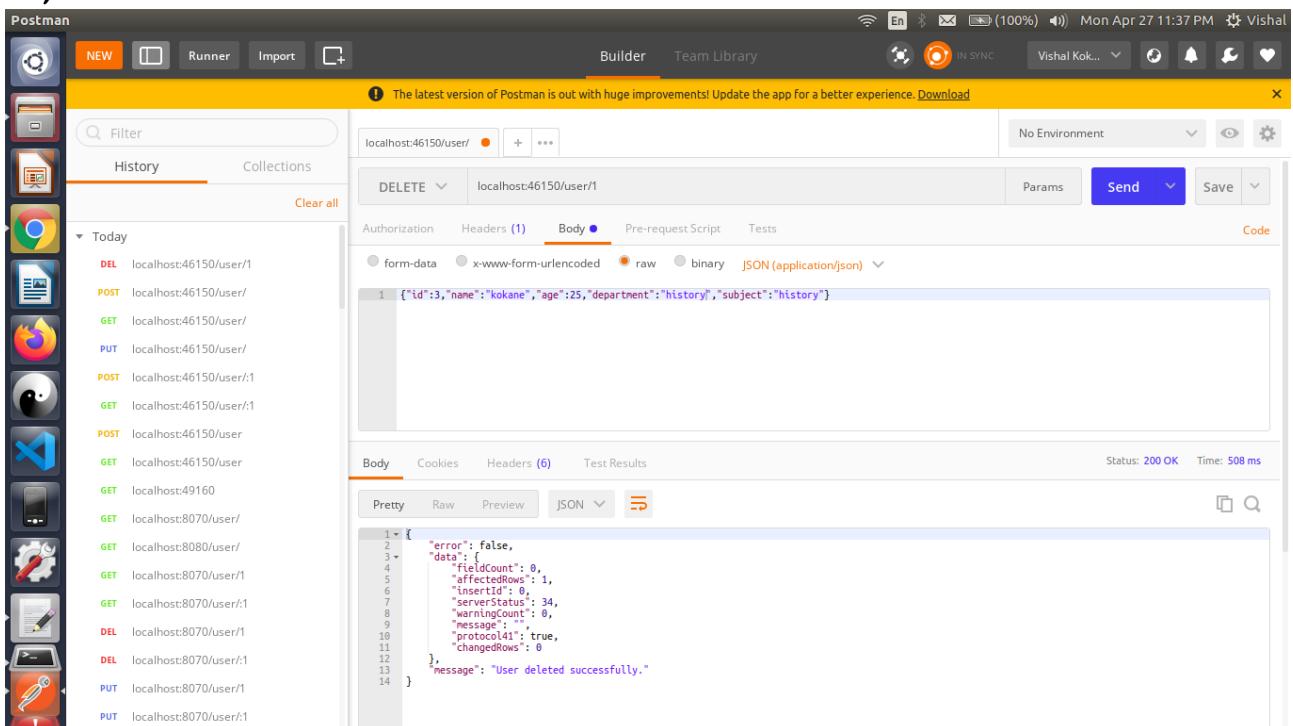


```
docker exec -it 23aad069902a /bin/bash
[docker exec -it 23aad069902a /bin/bash] docker exec -it 23aad069902a /bin/bash [vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/...]
mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

6) Postman delete data with user id =1



The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons and a list of recent requests. The main area has a yellow header bar with a message about the latest version. Below it, there's a search bar and tabs for 'History' and 'Collections'. The main workspace shows a 'DELETE' request to 'localhost:46150/user/1'. The 'Body' tab is selected, showing a JSON payload:

```
{"id":3,"name":"kokane","age":25,"department":"history","subject":"history"}
```

. The 'Params' tab is empty. To the right, there are 'Send' and 'Save' buttons. At the bottom, the response is shown with a status of '200 OK' and a time of '508 ms'. The response body is a JSON object:

```
1  {
2      "error": false,
3      "data": {
4          "fieldCount": 0,
5          "affectedRows": 1,
6          "insertId": 0,
7          "serverStatus": 34,
8          "warningCount": 0,
9          "message": "",
10         "protocol41": true,
11         "changedRows": 0
12     },
13     "message": "User deleted successfully."
14 }
```

6.2) Mysql status

```
docker exec -it 23aad069902a /bin/bash
mysql> select * from userdb.user;
+----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

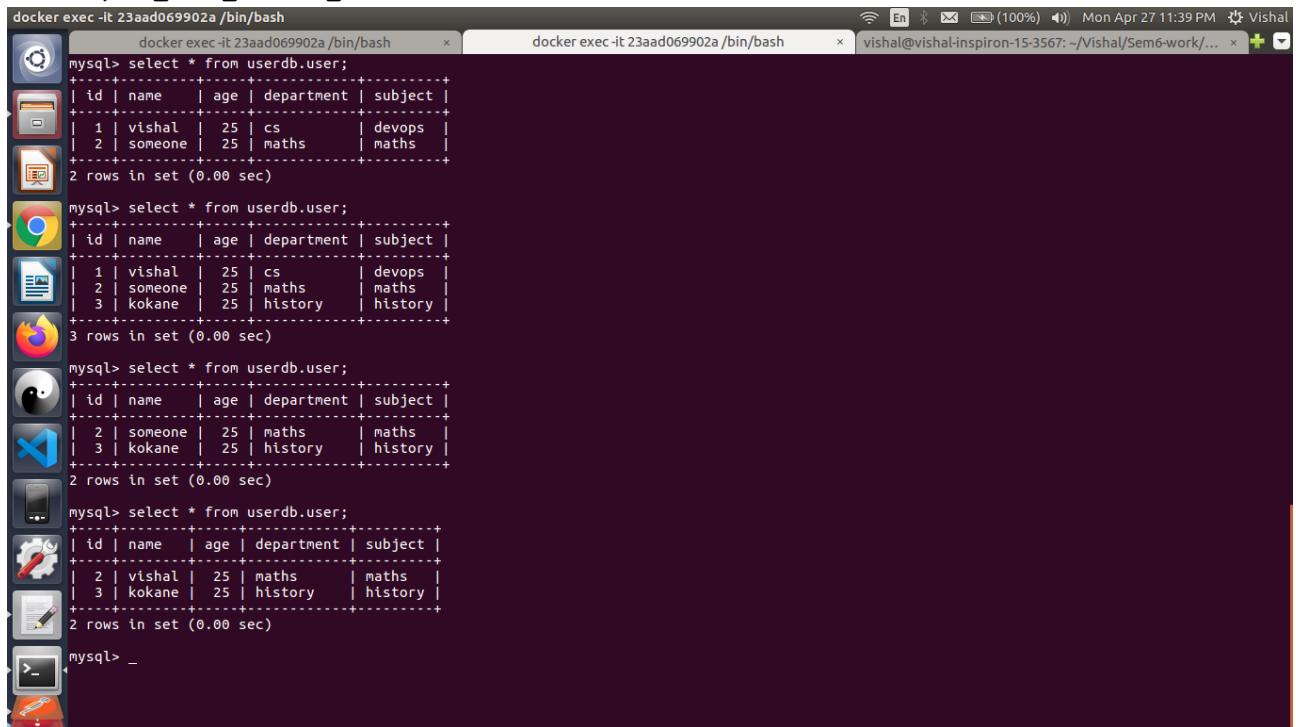
7) Postman PUT :update user name using userid

The screenshot shows the Postman application interface. On the left, there's a sidebar with a history of API calls and a central workspace for building requests. In the main workspace, a PUT request is being prepared to the URL `localhost:46150/user/`. The request body is set to raw JSON:

```
{"id":2,"name":"vishal","age":25,"department":"maths","subject":"maths"}
```

The response pane at the bottom shows a successful 200 OK status with a response message indicating the user was updated successfully.

7.2) Mysql update status



The screenshot shows a Linux desktop environment with several open windows. In the top right corner, there is a system tray with icons for battery (100%), signal strength, and volume. The date and time are displayed as 'Mon Apr 27 11:39 PM'. Below the tray, there is a window titled 'vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/...' which contains the MySQL command-line interface. The MySQL session shows three separate 'select * from userdb.user;' queries being run, each returning different sets of data. The first query returns 2 rows, the second returns 3 rows, and the third returns 2 rows. The data columns are id, name, age, department, and subject.

```
mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1  | vishal | 25 | cs       | devops   |
| 2  | someone | 25 | maths    | maths    |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1  | vishal | 25 | cs       | devops   |
| 2  | someone | 25 | maths    | maths    |
| 3  | kokane  | 25 | history  | history  |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 2  | someone | 25 | maths    | maths    |
| 3  | kokane  | 25 | history  | history  |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 2  | vishal | 25 | maths    | maths    |
| 3  | kokane  | 25 | history  | history  |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

8) Postman get all records

The screenshot shows the Postman application interface. In the top navigation bar, there are buttons for NEW, Runner, Import, and a search bar. To the right, it shows the date and time as Mon Apr 27 11:33 PM and the user's name as Vishal. The main workspace has a yellow banner at the top stating "The latest version of Postman is out with huge improvements! Update the app for a better experience. Download". Below this, there is a sidebar titled "History" which lists various API requests made today, including GET, PUT, POST, and DELETE methods on the localhost:46150/user endpoint. The main panel shows a "GET" request to "localhost:46150/user/" with an orange status indicator. The "Authorization" tab is selected, showing "No Auth". The "Body" tab is selected, displaying a JSON response: [{"id":1,"name":"vishal","age":25,"department":"cs","subject":"devops"}, {"id":2,"name":"someone","age":25,"department":"maths","subject":"maths"}]. The status bar at the bottom indicates "Status: 200 OK" and "Time: 37 ms".

9) Postman delete fail record not exist

This screenshot shows the same Postman interface as the previous one, but with a successful DELETE operation. The history sidebar shows a "DELETE" request to "localhost:46150/user/13". The main panel shows a "DELETE" request to "localhost:46150/user/13" with an orange status indicator. The "Authorization" tab is selected, showing "No Auth". The "Body" tab is selected, displaying a JSON response: {"error": false, "data": {}, "affectedRows": 0, "insertId": 0, "serverStatus": 34, "warningCount": 0, "message": "Protocol41", "changedRows": 0}. The status bar at the bottom indicates "Status: 200 OK" and "Time: 43 ms".

10) Postman delete Success

Postman

The latest version of Postman is out with huge improvements! Update the app for a better experience. [Download](#)

Builder Team Library

localhost:46150/user/

DELETE localhost:46150/user/2

Params Send Save

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON

Status: 200 OK Time: 615 ms

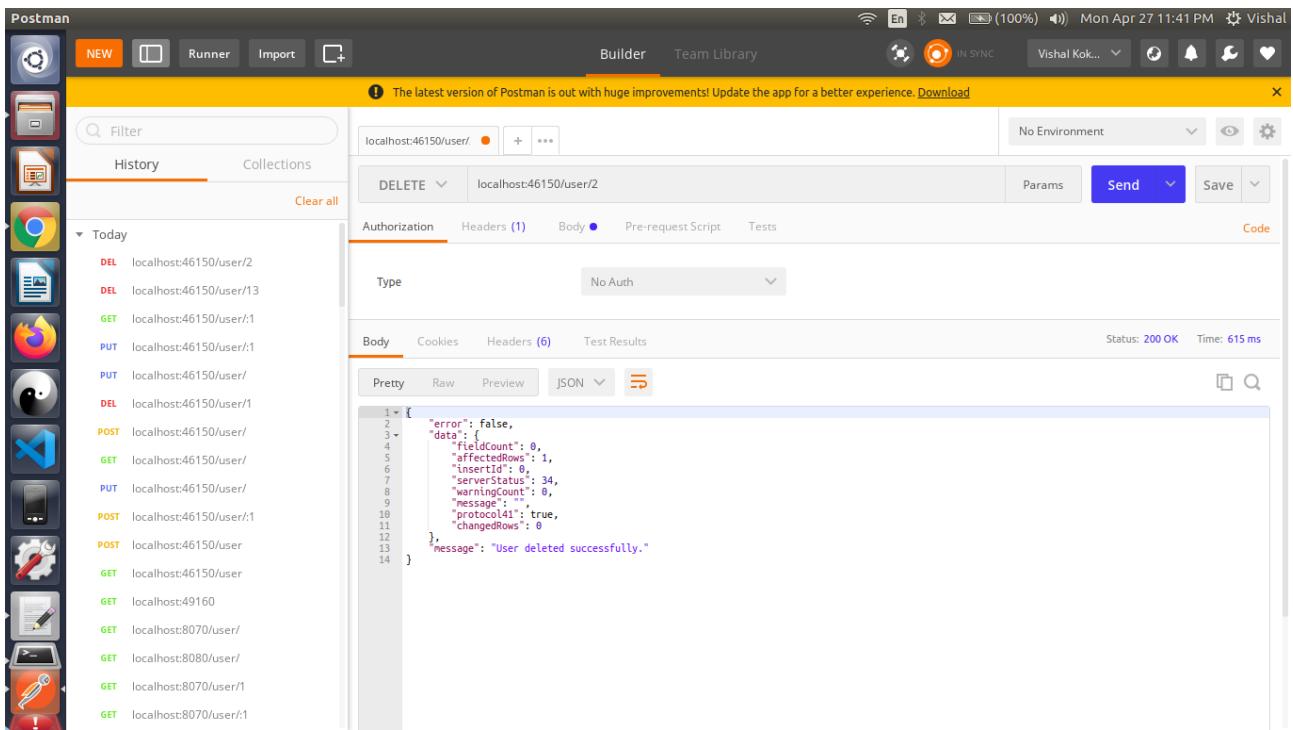
Code

History Collections Clear all

Today

- DEL localhost:46150/user/2
- DEL localhost:46150/user/13
- GET localhost:46150/user/:1
- PUT localhost:46150/user/:1
- PUT localhost:46150/user/
- DEL localhost:46150/user/1
- POST localhost:46150/user/
- GET localhost:46150/user/
- PUT localhost:46150/user/
- POST localhost:46150/user/:1
- POST localhost:46150/user
- GET localhost:46150/user
- GET localhost:49160
- GET localhost:8070/user/
- GET localhost:8080/user/
- GET localhost:8070/user/:1
- GET localhost:8070/user/:1

```
1 < [  
2   "error": false,  
3   "data": {  
4     "fieldCount": 0,  
5     "affectedRows": 1,  
6     "insertId": 0,  
7     "serverStatus": 34,  
8     "warningCount": 0,  
9     "message": "",  
10    "protocol41": true,  
11    "changedRows": 0  
12  },  
13  "Message": "User deleted successfully."  
14 ]
```



11) Postman Get by id

The screenshot shows the Postman application interface. The left sidebar displays a history of requests, including several GET requests to localhost:46150/user/. The main workspace shows a GET request to localhost:46150/user/:3. The response status is 200 OK with a time of 72 ms. The response body is a JSON array containing one element: [{"id":3,"name":"kokane","age":25,"department":"history","subject":"history"}].

```
[{"id":3,"name":"kokane","age":25,"department":"history","subject":"history"}]
```